The Bayesian Committee Support Vector Machine

Anton Schwaighofer[1,2] and Volker Tresp[2]

[1] TU Graz, Institute for Theoretical Computer Science
Inffeldgasse 16b, 8010 Graz, Austria
aschwaig@igi.tu-graz.ac.at
http://www.igi.tu-graz.ac.at/aschwaig/
[2] Siemens AG, Corporate Technology, Otto-Hahn-Ring 6,
81739 München, Germany
{Anton.Schwaighofer.external,Volker.Tresp}@mchp.siemens.de

Abstract. Empirical evidence indicates that the training time for the support vector machine (SVM) scales to the square of the number of training data points. In this paper, we introduce the Bayesian committee support vector machine (BC-SVM) and achieve an algorithm for training the SVM which scales linearly in the number of training data points. We verify the good performance of the BC-SVM using several data sets.

1 Introduction

Kernel-based systems such as the support vector machine (SVM) and Gaussian processes (GP) are powerful and currently very popular approaches to supervised learning [1]. Unfortunately, there are at least three problems when one tries to scale up these systems to large data sets. First, training time increases drastically with the number of training data points, second, memory requirements increase with data set size and third, prediction time is proportional to the number of kernels and the latter is equal to (or at least increases with) the number of training data points.

One approach to scale up kernel systems to large data sets is the Bayesian committee machine (BCM [6]). In the BCM the data set is divided up into M sets of approximately the same size and M models are developed on the individual data sets. The predictions of the individual models are combined using a weighting scheme which is derived from a Bayesian perspective. The computational complexity of the BCM scales linearly in the number of training data points. The BCM was developed in the context of Gaussian process regression (GPR) and was later applied to generalized Gaussian processes regression (GGPR). GGPR uses measurement processes derived from the exponential family of distributions and can be applied to classification and the prediction of counts and lifetimes.

In this paper we apply the BCM approximation to the support vector machine (SVM) in form of the Bayesian committee support vector machine (BC-SVM).

The SVM is a kernel-based classifier which is derived from VC learning theory [1] and minimizes a bound on the generalization error. A particular property of the SVM is that it achieves sparse solutions, i.e. many kernel weights are equal to zero. The basis for the applicability of the BCM to the SVM was made possible by a recent probabilistic interpretation of the SVM by Sollich [4].

The paper is organized as follows. In the next sections we will review Gaussian process regression, the BCM and the SVM. In Section 5 we will derive the BC-SVM. In Section 6 we present experimental results and in Section 7 we present conclusions.

2 Gaussian Process Regression (GPR)

In contrast to the usual parameterized approach to regression, in Gaussian process regression we specify the prior model directly in function space. In particular we assume that *a priori* f is Gaussian distributed (in fact it would be an infinite-dimensional Gaussian) with zero mean and a covariance $K(x_1, x_2) = cov(f(x_1), f(x_2))$, where the experimenter has to specify the covariance function.

We assume that we can only measure a noisy version $y(x) = f(x) + \epsilon(x)$ of the underlying function f, where $\epsilon(x)$ is independent Gaussian distributed noise with zero mean and variance σ_ψ^2.

The goal is now to predict the functional values $f^q = (f_1^q, \ldots, f_{N_Q}^q)'$ at a set of N_Q test or query points based on a set of N training data $\mathcal{D} = \{(x_1, y_1), \ldots (x_N, y_N)\}$. Let $\Psi^{mm} = \sigma_\psi^2 I$ be the noise variance of the targets of the measurements, where I is the unit matrix. Furthermore, Σ^{mm} is the covariance matrix of the functional values at the training data points, Σ^{qq} is the covariance matrix at the query points and Σ^{qm} is the covariance matrix between training points and the query points.

Under these assumptions the conditional density of the response variables at the query points is Gaussian distributed with mean $E(f^q|\mathcal{D})$ and covariance $cov(f^q|\mathcal{D})$, given by

$$E(f^q|\mathcal{D}) = \Sigma^{qm}(\Psi^{mm} + \Sigma^{mm})^{-1} y^m \quad (1)$$

$$cov(f^q|\mathcal{D}) = \Sigma^{qq} - \Sigma^{qm} \, (\Psi^{mm} + \Sigma^{mm})^{-1} \, (\Sigma^{qm})'. \quad (2)$$

Note, that following Eq. 1, the estimated function at input x becomes

$$\hat{f}(x) = \sum_{i=1}^{N} w_i K(x, x_i). \quad (3)$$

where $w = (\Psi^{mm} + \Sigma^{mm})^{-1} y^m$. That is, the prediction of the GPR system is a weighted combination of kernels on the training data points.

3 The Bayesian Committee Machine Applied to GPR

The calculation of the optimal predictions at the query points (Eq. 1) requires the inversion of an $N \times N$ matrix and therefore scales to the third power of the number of training data points. As a consequence, GPR is limited to problems up to may be 1000 training data points. In the Bayesian Committee Machine (BCM, [6]), the data are partitioned into M data sets $\mathcal{D} = \{D^1, \ldots, D^M\}$ (of typically approximately same size) and M GPR systems are trained on the individual data sets D^i. When applied to a set of query points, each of the M GPR systems outputs a prediction $E(f^q|D^i)$ together with covariance $cov(f^q|D^i)$, calculated employing GPR formulas Equations 1 and 2.

The BCM combines the M estimates and calculates an approximation to the expected values $\hat{E}(f^q|\mathcal{D})$ of the functional values at the query points as

$$\hat{E}(f^q|\mathcal{D}) = C_{BCM}{}^{-1} \sum_{i=1}^{M} cov(f^q|D^i)^{-1} E(f^q|D^i) \tag{4}$$

$$C_{BCM} = \widehat{cov}(f^q|\mathcal{D})^{-1} = -(M-1)(\Sigma^{qq})^{-1} + \sum_{i=1}^{M} cov(f^q|D^i)^{-1}. \tag{5}$$

We recognize that the prediction of each GPR system i is weighted by the inverse covariance of its prediction. In [6] it was shown that the BCM approximation becomes equal to the correct expected value when the number of query points is equal to the effective number of parameters of the kernel system. If we set $N/M = N_Q$ (i.e. the number of data points in each module is equal to the number of query points which is a typical choice) then the computational complexity of the BCM scales as $\mathcal{O}(N \times N_Q^2)$, i.e. linear in the training data set size.

4 The Support Vector Machine (SVM)

The SVM is typically formulated as a linear classifier which is applied to a high-dimensional feature space using the kernel transformation. The SVM takes on the form of a kernel system as in Eq. 3 and a new pattern is classified according to the sign of $\hat{f}(x) + b$ where b is a constant. The optimal weight vector w is found by minimizing the cost function

$$cost = \frac{1}{2} w' \Sigma^{mm} w + \frac{C}{2} \sum_{i=1}^{N} \left[(1 - y_i(f(x_i) + b))_+\right]^{\alpha}.$$

Here, $y \in \{-1, 1\}$ is the class label and, as before, $(\Sigma^{mm})_{ij} = K(x_i, x_j)$ is the kernel matrix. The operation $()_+$ sets all negative values equal to zero and C is a constant that determines to what degree a violation of the margin constraints is penalized. In most applications $\alpha = 1$ has been used (1-norm soft margin) but $\alpha = 2$ (2-norm soft margin) is also a common choice [1].

The state of the art optimization routines (SMO, SVM^{light} [1]) scale approximately $\mathcal{O}(N^2)$, which limits the applicability of the SVM to data sets up to a few 10000 training data points.

Sollich [4] has shown how the SVM can be formulated in the context of Gaussian processes. As in GPR, we assume that the function is derived from an infinite dimensional prior Gaussian distribution. In the SVM, we use a different measurement process[1]. Instead of assuming Gaussian measurement noise, we set

$$P(y|f(x)) \propto \exp\left(-\frac{C}{2}\left[(1-y(f(x)+b))_+\right]^\alpha\right).$$

The cost function that is minimized during SVM training can then be interpreted as the negative log of the posterior probability $-\log P(f^m|\mathcal{D})$ and becomes

$$-\log P(f^m|\mathcal{D}) = \frac{1}{2}(f^m)'(\Sigma^{mm})^{-1}f^m + \frac{C}{2}\sum_{i=1}^{N}\left[(1-y_i(f(x_i)+b))_+\right]^\alpha . \qquad (6)$$

Here, f^m is the vector of SVM outputs $f(x_i)$ at the training data points x_i.

5 The Bayesian Committee Support Vector Machine (BC-SVM)

For deriving the BC-SVM, we assume that the posterior density $P(f^m|\mathcal{D})$ can be approximated by a Gaussian density (Laplace approximation). The mean of the Gaussian corresponds to the prediction of the SVM at these points and can be calculated using Eq. 3. The covariance matrix is derived from the Hessian matrix of the cost function. We make use of Sollich's re-formulation of the SVM cost function $-\log P(f^m|\mathcal{D})$ (Eq. 6).

Note, that the SVM cost function with $\alpha = 1$ is non-differentiable at the points with $1 - yf(x) = 0$. This is particularly unpleasant since the points with the property $1 - yf(x) = 0$ are the support vectors. Sollich [4] proposes a Gaussian approximation in this case but we experienced instabilities when we used this approximation. For this reason, we used $\alpha = 2$ in our experiments. Here, the second derivative of the likelihood term is equal to C for the support vectors and is 0 otherwise.

Computing the BC-SVM approximation thus consists of the following steps:

1. Train SVM module i on partition $\mathcal{D}^i$ of the training data, using some standard SVM training algorithm.
2. The SVM outputs at the query points, as given by Eq. 3, replace $E(f^q|\mathcal{D}^i)$ in Eq. 4.
3. Compute the Hessian matrix H of the cost function Eq. 6 evaluated at the training points. By analogy to GPR, we calculate $cov(f^q|\mathcal{D}^i)$ (Eq. 2) by identifying $H = (\Sigma^{mm})^{-1} + (\Psi^{mm})^{-1}$.
4. Use the predictions and their covariances $cov(f^q|\mathcal{D}^i)$ of all modules in the BCM (Eq. 4) to obtain the prediction of the BC-SVM.

[1] Properly normalizing the conditional probability density is somewhat tricky and is discussed in detail in [4].

For the BC-SVM Eq. 2 further simplifies such that the covariances are defined only for the support vectors: Σ^{mm} contains the covariances between only the support vectors, Σ^{qm} contains the covariances between the support vectors and the query points, and $\Psi^{mm} = \frac{1}{C} I$ where I is the unit matrix with the dimensionality of the number of support vectors.

All 2-norm SVMs have been trained using adapted versions of the decomposition method and working set selection in the style of SVM^{light}, as proposed by Joachims [2].

6 Experiments

We demonstrate the performance of the BC-SVM in experiments on four artificial data sets (with different levels of noise) and on four real world data sets.

The ART data sets were generated by randomly drawing $d = 5$ dimensional input points uniformly from $[-1, 1]^d$, using them as inputs to a map defined by 5 normalized Gaussian basis functions, and finally adding independent Gaussian noise with variance σ_ψ^2 to the map. The assigned class corresponds to the sign of the generated outputs. In this way, the degree of overlap between classes can be influenced by varying σ_ψ^2. (This is the same data set as used in [7], see the reference for a detailed description.)

The CHECKER data set is sampled uniformly from a 4×4 checkerboard grid on $[0, 1] \times [0, 1]$. CHECKER is often used to demonstrate highly complex decision surfaces. Note that data sets CHECKER and ART with $\sigma_\psi = 0$ have non-overlapping classes.

Table 1 shows test set errors for different classification methods on those data sets. Shown are results for an SVM trained on the whole of the training data and for BC-SVMs with different module and query set size. It can be seen that the BC-SVM, even with a small module size such as 100, performs comparable to a full SVM on all data sets with noise (ART with $\sigma_\psi > 0$). For the noise free sets CHECKER and ART with $\sigma_\psi = 0$, a larger module size is required to obtain a performance that is comparable with the full SVM.

The evolution of the test set error, as the number of training points (and thus the number of modules in the BC-SVM) increases, is shown in Figure 1 for ART $\sigma_\psi = 0.1$. The graph furthermore shows that the combination scheme employed by the BC-SVM performs significantly better than averaging the predictions of the individual modules.

The performance of the BC-SVM shows up more clearly in Table 2, where the results for four real world data sets are given. For the three smaller data sets PIMA, BUPA and WAVEFORM, the BC-SVM performs comparable to a full SVM and no statistically significant difference in performance can be seen. For the large ADULT data set, the BC-SVM has a slightly higher error rate than the full SVM, yet still performs excellently.

For the ADULT data set we have also logged the training time. Training the BC-SVM takes about 3 minutes, while training the full SVM lasts a few hours. Both SVM and BC-SVM have been implemented in Matlab. (Comparing

Table 1. BC-SVM on 4 artificial data sets. Shown is the number of input dimensions d, the number of training examples N, and the average test set error for different classification methods. BC-SVM(i, j) uses a module size i with query set size j. All test set errors have been computed on independent 10000 point test sets.

	ART $\sigma_\psi = 0$	ART $\sigma_\psi = 0.1$	ART $\sigma_\psi = 0.5$	CHECKER
d	5	5	5	2
N	2000	2000	2000	5000
SVM	2.58%	7.25%	19.32%	1.26%
BC-SVM(100,200)	3.30%	7.87%	19.75%	2.67%
BC-SVM(200,200)	3.54%	7.44%	19.54%	2.65%
BC-SVM(500,200)	—	—	—	1.82%

Table 2. BC-SVM on 3 real-world data sets. Shown is the number of input dimensions d, the number of training examples N, and the average test set error for different classification methods. For BUPA, PIMA and WAVEFORM, test set errors have been computed using 7fold cross-validation, and are given together with a 95% confidence interval. The results for ADULT are on an independent test set of 16281 points.

	BUPA	PIMA	WAVEFORM	ADULT
d	6	8	21	124
N	345	700	2000	32561
SVM	$27.7 \pm 2.7\%$	$24.3 \pm 2.7\%$	$8.33 \pm 0.99\%$	14.77%
BC-SVM(50,50)	$27.1 \pm 3.7\%$	$24.4 \pm 2.9\%$	$8.33 \pm 0.89\%$	—
BC-SVM(100,100)	$27.4 \pm 3.9\%$	$25.7 \pm 2.6\%$	$8.94 \pm 1.5\%$	16.88%
BC-SVM(200,200)	—	—	$8.66 \pm 0.98\%$	15.76%

the BC-SVM runtime directly with methods such as SMO and SVM^{light} would require a C/C++ implementation.)

It should be mentioned that predictions of the BC-SVM on test data are more time consuming than predictions of a standard SVM. This results from the matrix operations in the combination scheme (Eq. 4), and from the higher total number of support vectors in the BC-SVM. The modules in a BC-SVM, trained on small portions of the data, can not exploit sparseness as efficiently as an SVM that has been trained on the full data.

7 Conclusions

We have shown a principled way of scaling support vector machines to large data sets by training SVMs on small subsets of the training data and combining their predictions through a Bayesian scheme. The module size in this BC-SVM is small enough that they can be trained with off-the-shelf quadratic optimizers without the necessity for highly optimised SVM training methods (decomposition, SMO, etc). We found the predictions of the BC-SVMs to be comparable with standard SVMs trained on the full data.

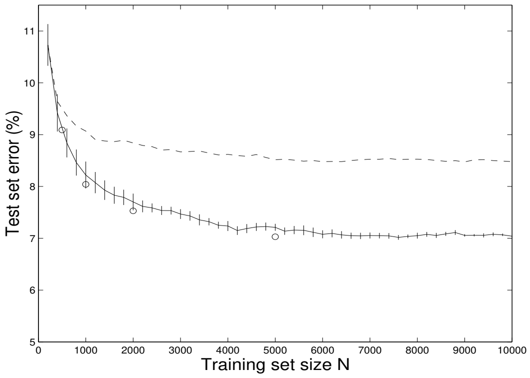

Fig. 1. Artificial data with noise $\sigma_\psi = 0.1$: Test set error of SVM compared to BC-SVM(200,200) as a function of the number of training points N. The test set errors of the SVM's, trained on various training set sizes, are shown as circles. The solid line is the error of the BC-SVM, together with (two-sided) 95% confidence intervals. The BC-SVM does not perform significantly worse than the full SVM. In particular, the BC-SVM clearly outperforms a combination scheme that averages the predictions of the BC-SVM modules (shown dashed).

An interesting direction for further work would be to apply the BCM to relevance vector machines (RVM [5]). The RVM uses a probabilistic approach to obtain a sparse solution of a kernel system. So far, the RVM idea is lacking a method for scaling up to large data sets, since decomposition methods are not applicable.

Acknowledgements AS gratefully acknowledges support through an Ernst-von-Siemens scholarship.

References

1. Cristianini, N., Shawe-Taylor, J.: Support Vector Machines. Cambridge University Press, 2000
2. Joachims, T.: Making Large-scale Support Vector Machine Learning Practical. In: Schölkopf, B., Burges, C.J., Smola, A.J. (eds.), *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1998 pages 169–184
3. Solla, S.A., Leen, T.K., Müller, K.R. (eds.): Advances in Neural Information Processing Systems 12. MIT Press, 2000
4. Sollich, P.: Probabilistic Methods for Support Vector Machines. In: Solla et al. [3], pages 349–355, 2000
5. Tipping, M.E.: The Relevance Vector Machine. In: Solla et al. [3], 2000
6. Tresp, V.: A Bayesian Committee Machine. Neural Computation 12 (2000) 2719–2741
7. Tresp, V.: The Generalized Bayesian Committee Machine. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, MA USA, 2000